第16章 Linux 内核管理与程序开发基础

16.1 学习引导

【单元概述】

在用户安装完 Red Hat Enterprise Linux 或者其他版本的 Linux 之后，某些用户也许需要重新对系统内核进行配置和重建 Linux 内核。Red Hat Linux 带有预先构建好的系统内核，这样可以尽量简化安装过程，但这种事先构建好的内核可能没有得到正确的配置，不能充分利用系统提供的各项功能。用户通过对内核的重新配置和重建，可以得到一个为自己所需要的系统而定制的内核。

在 Linux 应用程序开发、内核开发及驱动开发中，C/C++语言是应用最为广泛的开发语言，因为 Linux 的内核就是 C 语言程序。但是，在 Linux 下进行程序开发的难度在一定程度上要高于在 Windows 下进行程序开发，因此用户如果要自己对内核进行重新编译，必须掌握在 Linux 下进行程序开发的相关基础知识。

本章首先介绍 Linux 内核管理，内核各部分关系以及 Linux 下的内核文件功能。同时，将详细介绍如何对 Redhat Linux 内核进行升级。其次主要介绍 Linux 程序开发基础，重点在于 Linux 操作系统下 C 语言的编译调试工具 GCC、GDB 和G++的使用，最后以一个实例向读者介绍编译 C 程序的基本过程。为读者进一步学习打下良好的基础。

【知识要点及掌握程度】

理解 Linux 的结构和主要功能，结合操作系统的相关原理知识，深入了解进程调试和内存管理等相关知识。

掌握 Linux 内核的重建方法，掌握获取源代码、配置和编译 Linux 内核、安装新编译的内核等具体的操作方法。

掌握 Linux 操作系统下 C 语言的编译调试工具 GCC、GDB 和G++的使用方法。

通过系统内核编译实例与 C 程序项目的编译过程深入理解本章所学知识并能熟练运用相关的工具。

【教学重点与难点】

重点:内核的重建、程序开发工具 GCC、GDB、G++的使用方法。

难点:内核的重建。

【教学设计与实施方法】

理论与项目实践相结合,通过项目案例巩固本章重点难点内容。

【实践环节设计】

单元项目:Linux 内核的重建。

【教学效果测评】

合格:在教师指导下完成习题 1、2;

良好:在教师指导下完成习题 1,独立完成习题 2;

优秀:独立完成习题 1、2。

16.2 Linux 内核管理

16.2.1 内核主要功能

在 Linux 操作系统中,/usr/src 目录专门用于存放 Linux 源程序,即 Linux 内核源代码程序。Linux 内核主要由以下 5 个子系统组成:进程调度、内存管理、虚拟文件系统、网络接口及进程间通信。

(1)进程调度(SCHED):控制进程对 CPU 的访问权限。当需要选择下一个进程运行时,由调度程序选择最需要运行的进程,此处的可运行进程是指实际上仅等待 CPU 资源的进程(不需要等待其他任何资源),如果某个进程在等待其他资源,则该进程是一个不可运行进程。Linux 使用了比较简单的基于优先级的进程调度算法选择新的进程。

(2)内存管理(MM):允许多个进程安全地共享主内存区域。Linux 的内存管理支持虚拟内存,即在计算机中运行的程序,其代码、数据、堆栈的总量可以超过实际内存的大小,操作系统只是把当前使用的程序块保留在内存中,其余的程序块则保留在磁盘中。必要时,操作系统负责在磁盘和内存间交换程序块。内存管理从逻辑上分为硬件无关部分和硬件相关部分。

硬件无关部分提供了进程的映射和逻辑内存的对换。

硬件相关的部分为内存管理硬件提供虚拟接口。

(3)虚拟文件系统(VirtualFileSystem,VFS):隐藏了各种硬件的具体细节,为所有的设备提供了统一的接口,VFS 提供了多达数十种不同的文件系统。虚拟文件系统可以分为逻辑文件系统和设备驱动程序。

逻辑文件系统是指 Linux 所支持的文件系统,如 ext2、fat 等。

设备驱动程序指为每一种硬件控制器所编写的设备驱动程序模块。

(4)网络接口(NET):提供了对各种网络标准的存取和各种网络硬件的支持。网络接口可分为网络协议和网络驱动程序。

网络协议部分负责实现每一种可能的网络传输协议。

网络设备驱动程序负责与硬件设备通信,每种硬件设备都有相应的设备驱动程序。

(5)进程间通信(IPC):支持进程间的各种通信机制。

处于中心位置的是进程调度,其他所有的子系统都依赖于它,因为每个子系统都需要挂起或恢复进程。

一般情况下,当一个进程等待硬件操作完成时,它被挂起;当操作真正完成时,进程被恢复执行。例如,当一个进程通过网络发送一条消息时,网络接口需要挂起发送进程,直到硬件完成消息的发送,当消息被成功的发送出去以后,网络接口给进程返回一个代码,表示操作的成功或失败。其他子系统以相似的理由依赖于进程调度。

16.2.2　各个子系统之间的依赖关系

(1)进程调度与内存管理之间的关系:这两个子系统互相依赖。在多道程序环境下,程序要运行必须先创建进程,而创建进程的第 1 件事情,就是分配内存空间,将此程序的程序代码和数据装入内存。

(2)进程间通信与内存管理的关系:进程间通信子系统要依赖内存管理所支持的共享内存通信机制,这种机制允许两个进程除了拥有属于自己的私有空间外,还可以存取共同的内存区域,从而完成数据交换。

(3)虚拟文件系统与网络接口之间的关系:虚拟文件系统可以利用网络接口所支持的网络文件系统(NFS),也可以利用内存管理支持 RAMDISK 类型的设备。

(4)内存管理与虚拟文件系统之间的关系:内存管理利用虚拟文件系统支持交换功能,交换进程(swapd)定期由调度程序调度,这也是内存管理依赖于进程调度的惟一原因。当一个进程存取的内存映射被换出时,内存管理将向文件系统发出请求,同时挂起当前正在运行的进程。

除了这些依赖关系外,内核中的所有子系统还要依赖于一些共同的资源。这些资源包括所有子系统都用到的进程。例如分配和释放内存空间的进程,打印警告或错误信息的进程,还有系统的调试进程等。

16.2.3　系统数据结构

在 Linux 内核的实现中,有一些数据结构使用频率较高。下面分别介绍这些数据结构。

(1)task_struct 数据结构:Linux 内核利用一个数据结构(task_struct)代表一个进程,代表进程的数据结构指针形成了一个 task 数组(Linux 中,任务和进程是相同的概念),这种指针数组有时也称为指针向量。这个数组的大小为 NR_TASKS(默认为 512),表明 Linux 系统中最多能同时运行的进程数目。

当建立新进程的时候,Linux 为新进程分配一个 task_struct 结构,然后将指针保存在 task 数组中。调度程序一直维护着一个 current 指针,用以指向当前正在运行的进程。

(2)Mm_struct 数据结构:每个进程的虚拟内存由一个 mm_struct 结构来代表,该结构实际

上包含了当前执行映像的有关信息，并且包含了一组指向 vm_area_struct 结构的指针，vm_area_struct 结构描述了虚拟内存的一个区域。

(3)Inode 数据结构：虚拟文件系统(VFS)中的文件、目录等均由对应的索引节点(inode)代表。每个 VFS 索引节点中的内容由文件系统专属的进程提供。VFS 索引节点只存在于内核内存中，实际保存于 VFS 的索引节点高速缓存中。如果两个进程用相同的进程打开，则可以共享 inode 的数据结构，这种共享通过两个进程中数据块指向相同的 inode 完成。

16.2.4 Linux 内核源代码的结构

Linux 内核源代码位于/usr/src/Linux 目录下。其主要包含以下几个主要的目录。

/include 子目录：包含了建立内核代码时所需的大部分包含文件，这个模块利用其他模块重建内核。

/init 子目录：包含了内核的初始化代码，这是内核工作的起点。

/arch 子目录：包含了所有与硬件结构相关的特定内核代码。Linux 支持多种微处理架构，如常见的 i386、alpha、ARM(2.6 以后版本)微处理器，分别用不同的文件夹来管理不同类型的处理器代码。

/drivers 子目录：包含了内核中所有常见的设备驱动程序，如块设备、SCSI 设备、CDROM 设备以及串口等。

/fs 子目录：包含了所有文件系统的代码。如：ext2/3、VFAT、JFFS、NTFS、FAT、USB 以及 PROC 等类型的文件系统。

/net 子目录：包含了内核中网络相关的代码，如 802、IPV4/6、IPX、IRDA、X25、ATM、ETHERNET 等。

/mm 子目录：包含了所有内存管理代码。

/ipc 子目录：包含了进程间通信代码，如信号量信息。

/kernel 子目录：包含了与主内核相关的代码。

下面简要介绍内核各部分功能：

(1)系统的启动和初始化：在基于 Intel 的系统上，当 loadlin.exe 或 LILO 把内核装入到内存并把控制权传递给内核时，内核开始启动。可查看 arch/i386/kernel/head.S 文件，head.S 文件主要进行特定结构的设置，然后跳转到 init/main.c 的 main( )文件执行。

(2)内存管理：内存管理的代码主要位于/mm 文件夹下。

特定结构的代码在 arch/*/mm 文件夹下。

缺页中断处理的代码在/mm/memory.c 文件中。

内存映射和页高速缓存器的代码在/mm/filemap.c 文件中。

缓冲器高速缓存在/mm/buffer.c 文件中。

交换高速缓存的代码在 mm/swap_state.c 文件和 mm/swapfile.c 文件中。

(3)内核相关：在内核中，与微处理器相关的特定结构代码存在 arch/*/kernel 文件夹下，调度相关的程序存放在 kernel/sched.c 文件中，fork( )相关的代码存入在 kernel/fork.c 文件中，内核例程处理程序存放在 include/Linux/interrupt.h 文件中，task_struct 数据结构存放在 inlucde/Linux/sched.h 文件中。

(4)进程间通信:所有的 System VIPC 对象权限都包含在 ipc_perm 数据结构中,可以在 include/Linux/ipc.h 文件中找到。

SystemV 消息在 ipc/msg.c 文件中实现。

共享内存在 ipc/shm.c 文件中实现。

信号量在 ipc/sem.c 文件中。

管道在/ipc/pipe.c 文件中实现。

(5)中断处理:内核的中断处理代码几乎是所有的微处理器特有的。中断处理代码存在 arch/i386/kernel/irq.c 中,其定义在 include/asm-i386/irq.h 中。

16.3　升级 Redhat Linux 内核项目

【项目描述】

假设当前 Linux 内核版本为 2.4.20-8,将当前内核升级到 2.6.18 版本

读者可以使用以下命令查看 Red Hat Linux 当前的内核版本:

```
[root@localhost root]# uname-r
2.4.20-8
```

【构思设计】

内核升级(重建)基本步骤如下:

(1)获取升级软件包;

(2)对升级软件包进行安装;

(3)配置和编译内核;

(4)安装新编译的内核及相关文件;

(5)重新启动完成升级。

【实施运行】

(1)升级软件包。

由于最新内容对部分软件组件有相应的需求,故需要首先升级部分依赖软件组件。操作步骤如下所示:

①下载最新的 Linux 内核源代码程序。

查看解压后的 README 文件,获取从 V2.4 到 V2.6 要升级更新的软件包。主要包括模块配置工具 module-init-tools(modules-init-tools-3.2.tar.bz2 及以上版本)和 mkinitrd(mkinitrd-4.1.18.1-1.i386.rpm)。mkinitrd 包依赖于 device-mapper 包,而 device-mapper 又依赖于 lvm2 包。要下载软件包,如下所示。

```
lvm2-2.00.25-1.01.i386.rpm//device-mapper 依赖软件包
device-mapper-1.00.19-2.i386.rpm//mkinitrd 依赖软件包
mkinitrd-4.1.18.1-1.i386.rpm//mkinitrd 软件包
```

```
linux-2.6.18.tar.gz//内核
modules-init-tools-3.2.tar.bz2//模块软件包
```

②首先安装 modules-init-tools-3.2.tar.bz2 软件包。

```
[root@localhost root]#tar-jxvf modules-init-tools-3.2.tar.bz2//解压文件
[root@localhost root]#cd modules-init-tools//进入解压后的目录
[root@localhost root]#./configure//生成 makefile 文件
[root@localhost root]# make moveold//修改以前的模块命令
[root@localhost root]# make all install//安装
[root@localhost root]# ./generate-modprobe.conf /etc/modprobe.conf
```

因为 modutils 3.0 和 Red Hat Enterprise Linuxs 自带的 modutils 2.4 版本不兼容，所以需要在 make 之前使用 make moveold 命令更新原来 Red Hat Enterprise Linuxs 自带的 modutils2.4。make moveold 命令将系统内原来的 lsmod、rmmod、insmod 和 modprobe 等文件分别改名为 lsmod.old、rmmod.old、insmod.old 和 modprobe.old。这样处理后，原有的 2.4 内核 modutils 还可以调用旧版本的 modutils 来工作。

由于新版本的 modutils 不再使用/etc/modules.conf，而是使用/etc/modprobe.conf，所以命令“./generate-modprobe.conf /etc/modprobe.conf”将原来的硬件设备配置文件/etc/modules.conf 转换成/etc/modprobe.conf。

③安装 mkinitrd 的最新版本，由于依赖关系，需要安装前面下载的 3 个软件包，步骤如下。

```
[root@localhost root]#rpm-ivh--nodeps lvm2-2.00.25-1.01.i386.rpm
[root@localhost root]#rpm-ivh device-mapper-1.00.19-2.i386.rpm
[root@localhost root]#rpm-ivh mkinitrd-4.1.18.1-1.i386.rpm
```

(2)编译内核。

这是升级内核的关键步骤，升级内核前需要进行配置，其目的是为了将需要的模块及驱动信息加入，而取消先前不需要的部分。配置内容命令主要有以下 3 个命令。

```
make config//最原始的命令，一条一条的选择
make xconfig//图形界面方式，很友好，但需要装图形界面组件支持
make menuconfig//命令行模式下的菜单配置
```

推荐使用第 2 和第 3 个命令。采用方法 3 来配置内核的步骤如下所示。

①将要编译的内核解压至/usr/src 目录下再进行配置。

```
[root@localhost up]# cp linux-2.6.18.tar.gz /usr/src/
[root@localhost up]# cd /usr/src/
[root@localhost src]# ls-l linux-2.6.18.tar.gz
-rw-r--r--    1 root     root     52467340 Dec  7 14:12 linux-2.6.18.tar.gz
[root@localhost src]# tar-xzvf linux-2.6.18.tar.gz
[root@localhost src]# cd linux-2.6.18
[root@localhost linux-2.6.18]# make menuconfig
```

等待一段时间后，将弹出图 16.1 所示的配置界面。

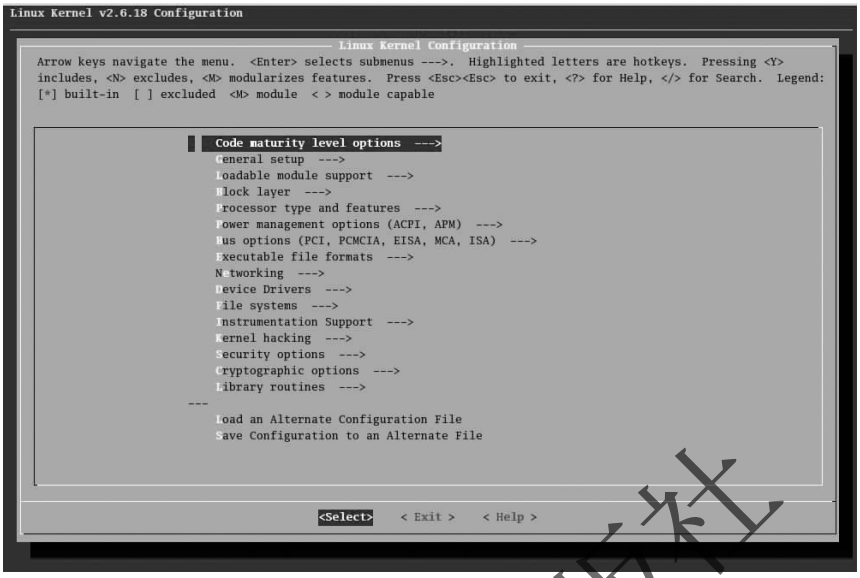

图 16.1　内核配置界面

图 16.1 中各选项内容如下所示。

Code maturity level options－－－>(代码成熟选项)。

[*]prompt for development add/or incomplete code/drives//系统默认选中且编译进内核

[*]Select only drivers expected com compile cleanly//系统默认选中且编译进内核

General setup－－－>(常规内核选项)。

[*]Support for paging of anonymous memory

[*]System V IPC

[*]POSIX Message Queues

[*]BSD Process Accounting

[*]Sysctl support

[*]Auditing support

[*]Support for hot-pluggable devices

[*]Kernel Userspace Events

Loadable module support－－－>(可加载模块支持)。

[*]Enable loadable module support

[*]Module unloading

[*]Automatic kernel module loading

Processor type and features－－－>(处理器类型和特性)。

Subarchitecture Type－－－>

该选项可以使 Linux 支持更多的 PC 标准,一般选 PC-compatible

Processor family－－－>

对 CPU 的类型进行选定,你是什么类型就选什么,也可以采用默认设置

[*]Generic X86 support

[*]HPET Timer Support

[*]Symmetric multi-processing support

[*]SMT(Hyperthreading) scheduler support

[*]MTRR (Memory Type Range Register) support

[*]Enable kernel irq balancing

Power management options(ACPI,APM)－－－>(高级电源管理)。

[*]Power Management support

[*]Software Suspend (EXPERIMENTAL)

<*> Processor

<*> Thermal Zone

Bus options(PCI,PCMCIA,EISA,MCA,ISA)－－－>(总线选项)。

[*]PCI support

[*]PCI device name database

Executable file formats－－－>(可执行文件格式)

[*]Kernel support for ELF binaries

<M>Kernel support for a.out and ECOFF binaries

<M>Kernel supprot for MISC binaries

Device Drivers－－－>(设备驱动程序)。

Generic Driver Options－－－>默认

[*] Select only drivers that don't need compile－time external firmware

[*] Prevent firmware from being built

Memory Technology Devices(MTD)－－－> 不选

Parallel port support－－－>并行端口,自定

Plug and Play support－－－>支持热插拔,自定

Block devices－－－>

<*>Normal floppy disk support

<M>Loopback device support

<M>Network block device support

<M>RAM disk support

IO Schedulers－－－>IO调度器

<*>Anticipatory I/O scheduler

<*>Deadline I/O scheduler

<*>CFQ I/O scheduler

ATA/ATAPI/MFM/RLL support－－－>ATA 设备,自定

SCSI device support－－－>SCSI 设备

<*>SCSI device support

[*]legacy /proc/scsi/ support

－－－SCSI support type (disk,tape,CD－ROM)

<*>SCSI disk support

Multi-device support (RAID and LVM)-－－>支持 RAID 和 LVM(逻辑卷),自定

[*] Multiple devices driver support (RAID and LVM)

<*>RAID support

<*>RAID-0 (striping) mode

<*>RAID-4/RAID-5 mode

<*>Multipath I/O support

＜＊＞Device mapper support

Fusion MPT device support－－－＞

＜＊＞Fusion MPT (base＋ScsiHost) drivers

＜＊＞Fusion MPT misc device (ioctl) driver

IEEE 1394 (FireWire) support－－－＞自定

I2O device support－－－＞自定

Networking support－－－＞网络选项

[＊] Networking support

Networking options－－－＞

＜＊＞Packet socket

＜＊＞Unix domain sockets 如果你有网络就选

[＊]TCP/IP networking

[＊]IP: TCP syncookie support (disabled per default)能防 DOS 攻击,但会降低一些性能,但性价比不错

＜＊＞IP: TCP socket monitoring interface

IP:Virtual Server Configuration－－－＞

[＊]Network packet filtering (replaces ipchains)－－－＞包过滤省略

[＊]Network device support

＜＊＞Bonding driver support－－－双网卡绑定的

ARCnet devices－－－＞

Ethernet (10 or 100Mbit)－－－＞

Ethernet (1000 Mbit)－－－＞

＜＊＞Broadcom Tigon3 support

ISDN subsystem－－－＞

Telephony Support－－－＞

Input device support－－－＞

Character devices－－－＞

I2C support－－－＞

Dallas's 1－wire bus－－－＞

Misc devices－－－＞

Multimedia devices－－－＞

Graphics support－－－＞

Sound－－－＞

USB support　－－－＞

MMC/SD Card support－－－＞

File systems－－－＞(文件系统)。

[＊]Ext2 extended attributes

[＊]Ext2 POSIX Access Control Lists

[＊]Ext2 Security Labels

＜＊＞Ext3 journalling file system support

[＊]Ext3 extended attributes

[＊]Ext3 POSIX Access Control Lists

[*]Ext3 Security Lables
[*]JBD (ext3) debugging support
[*]ReiserFS POSIX Access Control Lists
[*]Realtime support (EXPERIMENTAL)
[*]Quota support
[*]Security Label support
< * >Kernel automounter support
< * >kernel automounter version 4 support (also supports v3)
CD-ROM/DVD Filesystems-－－>
< * >ISO 9660 CDROM file system support
[*]Microsoft Joliet CDROM extensions
[*]Transparent decompression extension
<M>UDF file system support
DOS/FAT/NT Filesystems－－－>floppy要用到的文件格式
<M>Dos FAT fs support
<M>MSDOS fs support
<M>VFAT (windows-95) fs support
< * >NTFS file system support
[*]NTFS debugging support
[*]NTFS write support
Pseudo filesystems －－－>;
[*]/prco file system supprot
[*]Automatically mount at boot
[*]/dev/pts Extemded Attributes
[*]/dev/pts Security Labels
[*]Virtual memory file system support (former shm fs)
[*]HugeTLB file system support
Miscellaneous filesystems－－－>
Network file System－－－>
< * >NFS file system support
[*]Provide NFSv3 client supprot
[*]Provide NFSv4 client supprot (EXPERIMENTAL)
< * >NFS server support
< * >SMB file system support (to mount windows shares etc.)
[*]Use a default NLS
(cp437)Default Remote NLS Option (NEW)
Partition Types－－－>
[*]Advanced partition selection
[*]Acorn partition support
[*]Cumana partition support
[*]EESOX partition support
[*]ICS partition support

[*]Native filecore partition support
[*]PowerTec partition support
[*]RISCiX partition support
[*]Alpha OSF partition support
[*]Amiga partition table support
[*]Atari partition table support
[*]Macintosh partition map support
[*]PC BIOS (MSDOS partition tables) support
[*]BSD disklabel (FreeBSD partition tables) support
[*]Minix subpartition support
[*]Solaris (X86) partition table support
[*]Unixware slices support
[*]Windows Logical Disk Manager (Dynamic Disk) support
[*]Windows LDM extra logging
[*]NEC PC-9800 partition table support
[*]SGI partition support
[*]Ultrix partition table support
[*]Sun partition tables support
[*]EFI GUID Partition support
Native Language Support－－－>
Base native language support
(utf8) Default NLS Option－-可以修改让其支持中文
<M>Codepage 437 (United States ,Canada)
<M>Codepage 850 (Europe)
<M>Codepage 852 (Central/Eastern Europe)
<M>Simplified Chinese charset (CP936,GB2312)
<M>Traditional Chinese charset (Big5)
<M>Windows CP1250 (Slavic/Central European Languages)
<M>NLS ISO 8859-1 (Latin 1; Western European Languages)
<M>NLS UTF8
Profiling support－－－>。
Kernel hacking－－－>。
Security options－－－>。
Cryptographic options－－>。
Library routines－－－>。
< * >CRC32 functions
<M>CRC32c (Castagnoli,et al) Cyclic Redundancy-Check

在以上配置中,[*]直接编译进内核,< * >直接编译进内核,[M]编译成模块,<M>编译成模块,[ ]没被选中,< >没被选中,－－－>如果下面没有代码,则说明该项没被选用。完成以上配置后,保存退出。

②执行以下命令编译内核:

```
[root@localhost linux-2.6.18]# make
[root@localhost linux-2.6.18]# make zImage
```

③编译模块如下所示：

```
[root@localhost linux-2.6.18]#  make modules
[root@localhost linux-2.6.18]# make modules-install
```

(3)完成内核升级。

内核编译过程将有较长的一段时间，编译完成后，如果使用的是 GRUB，直接 reboot 就可以看到新内核选项。否则需要先将编译好的内核和相关文件复制到适合的目录中去，通常是根目录或者/boot。如果有 boot 分区，就将这些文件放到这个分区的根(/boot)中去，如果没有 boot 分区，就将这些文件放到根目录中去。

至此，内核升级(重建)操作完成。

16.4 Linux 下 C/C++开发工具简介

16.4.1 Linux 下 C 语言编译 GCC 工具

GCC 是 GNU 最优秀的自由软件之一，其主要提供 C/C++程序的编译、调试工作。在嵌入式 Linux 开发过程中，一般都采用 GCC 软件进行软件开发。本节将重点介绍如何使用 GCC 进行软件开发。

(1)GCC 版本信息。

GCC 是一种C++语言的编译器，它可以免费获得，其功能十分强大。可以在 http://ftp.gnu.org/gnu/gcc/的网站上找到正式的 Linux GCC 发布系统。

在“#”提示符号下键入 gcc-v，屏幕上就会显示出用户目前正在使用的 GCC 的版本。同时也可确定用户现在所用的是 ELF 格式还是 a.out 格式。

```
[root@yangzongde root]# gcc-v
Reading specs from /usr/lib/gcc-lib/i386-redhat-linux/3.2.2/specs
Configured with: ../configure --prefix=/usr --mandir=/usr/share/man--infodir=/usr/share/info--enable-shared--enable-threads=posix--disable-checking--with-system-zlib--enable--__cxa_atexit--host=i386-redhat-linux
Thread model: posix
gcc version 3.2.2 20030222 (Red Hat Linux3.2.2-5)
```

参数说明如下所示：

①i386 指明用户现在正在用的 gcc 是为 386 微处理器写的(用户电脑可能是 486 或是 586)。为这三种微处理器芯片编译而成的程序代码，彼此间是可以兼容使用的。

②redhat 标识当前 Linux 版本为 redhat。

③linux 其实指 linuxelf 或是 linuxaout。

④3.2.2 是版本的序号。

(2)GCC 目录结构

安装后，与 GCC 相关的几个重要目录分别如下所示。

①/usr/lib/gcc-lib/i386-redhat-linux/。

大部分的编译器在该文件目录下。除做编译可执行的程序工作的；另外，还有一些特定版本的程序库与头文件等也会保存在此。

②/usr/bin/gcc。

编译器的驱动程序，也就是用户实际在命令行上执行的程序。这个目录可供各种版本的 gcc 使用，只要用不同的编译器目录来安装就可以了。如果想强迫执行某个版本，键入 gcc-V version。例如：

```
[root@yangzongde root]# gcc-v
Reading specs from /usr/lib/gcc-lib/i386-redhat-linux/3.2.2/specs
Configured with: ../configure--prefix =/usr--mandir =/usr/share/man--infodir =/usr/share/
info--enable-shared--enable-threads=posix--disable-checking--with-system-zlib--enable-__cxa_atexit--
host=i386-redhat-linux
Thread model: posix
gcc version 3.2.2 20030222 (Red Hat Linux3.2.2-5)
[root@yangzongde root]# gcc-V2.6.3-v
Using built-in specs.
Configured with: ../configure--prefix =/usr--mandir =/usr/share/man--infodir =/usr/share/
info--enable-shared--enable-threads=posix--disable-checking--with-system-zlib--enable-__cxa_atexit--
host=i386-redhat-linux
Thread model: posix
gcc driver version 3.2.2 20030222 (Red Hat Linux3.2.2-5) executing gcc version 2.6.3
```

(3)GCC 执行过程。

gcc 编译器能将 C、C++语言源程序、汇编程序和目标程序编译连接成可执行文件，如果没有给出可执行文件的名字，gcc 将生成一个名为 a.out 的文件。在 Linux 系统中，可执行文件没有统一的后缀，系统从文件的属性来区分可执行文件和不可执行文件。而 gcc 则通过后缀来区别输入文件的类别。

①gcc 所遵循的部分约定规则如下所示：

后缀为.c 的文件：C 语言源代码文件。

后缀为.a 的文件：由目标文件构成的档案库文件。

后缀为.C、.cc 或.cxx 的文件：C++源代码文件。

后缀为.h 的文件：程序所包含的头文件。

后缀为.i 的文件：已经预处理过的 C 源代码文件。

后缀为.ii 的文件：已经预处理过的C++源代码文件。

后缀为.m 的文件：Objective-C 源代码文件。

后缀为.o 的文件：编译后的目标文件。

后缀为.s 的文件：汇编语言源代码文件。

后缀为.S 的文件：经过预编译的汇编语言源代码文件。

虽然 gcc 是 C 语言的编译器，但使用 gcc 由 C 语言源代码文件生成可执行文件的过程不仅仅是编译的过程。

②程序运行需要经历 4 个相互关联的步骤：

(i)预处理(也称预编译，Preprocessing)：命令 gcc 首先调用 cpp 进行预处理，在预处理过程中，对源代码文件中的文件包含(include)、预编译语句(如宏定义 define)进行分析。

(ii)编译(Compilation)：接着调用 cc1 进行编译，这个阶段根据输入文件生成以.o 为后缀的目标文件。

(iii)汇编(Assembly)：汇编过程是针对汇编语言的步骤，调用 as 进行工作，.S 为后缀的汇编语言源代码文件、.s 为后缀的汇编语言文件经过预编译和汇编之后都生成以.o 为后缀的目标文件。

(iv)链接(Linking)：当所有的目标文件都生成之后，gcc 就调用 ld 来完成最后的关键性工作，这个阶段就是连接。在连接阶段，所有的目标文件被安排在可执行程序中的恰当位置，同时，该程序所调用到的库函数也从各自所在的档案库中连到合适的地方。

(4)GCC 的基本用法和选项。

在使用 gcc 编译器的时候，必须给出一系列必要的调用参数和文件名称。gcc 编译器的调用参数大约有 100 多个，其中多数参数使用频率低，这里只介绍最基本、最常用的参数。

gcc 最基本的用法如下：

gcc [options] [filenames]

其中 options 就是编译器所需要的参数；filenames 给出相关的文件名称。常用参数如下：

-c：只编译，不链接成为可执行文件，编译器只是由输入的.c 等源代码文件生成.o 为后缀的目标文件，通常用于编译不包含主程序的子程序文件。

-o：output_filename，确定输出文件的名称为 output_filename，同时这个名称不能和源文件同名。如果不给出这个选项，gcc 就给出预设的可执行文件 a.out。

-g：产生符号调试工具(GNU 的 gdb)所必要的符号资讯，要想对源代码进行调试，就必须加入这个选项。

-O：对程序进行优化编译、连接，采用这个选项，整个源代码会在编译、连接过程中进行优化处理，这样产生的可执行文件的执行效率可以提高，但是，编译、连接的速度就相应地要慢一些。

-O2：比-O 更好的优化编译、连接，整个编译、连接过程会更慢。

-I dirname：将 dirname 所指出的目录加入到程序头文件目录列表中，是在预编译过程中使用的参数。

-E：生成.i 文件。让 gcc 在预处理后停止编译，从而生成.i 文件，此文件中包含所有预处理信息。

(5)G++编译器简介。

Gcc 中包含专用于C++程序编译的G++程序，该编译器能够读取并编译任何C++程序，在编译时，程序员需要使用的命令如下：

```
G++ -c *.cpp
```

常见的C++源程序扩展名有.C(大写)、.cc、.cxx。

16.4.2 GDB 调试工具

GNU 的调试器称为 gdb，该程序是一个交互式工具，工作在字符模式。在 X Window 系统

中，有一个 gdb 的前端图形工具，称为 xxgdb。在 Unix/Linux 环境下开发软件，gdb 比 VC、VB 具有更为强大的功能。

Gdb 做为功能强大的调试程序，可完成设置断点、监视程序变量的值、程序的单步执行、修改变量的值等功能。

在可以使用 gdb 调试程序之前，在编译源文件时必须使用-g 选项(即 gcc - c - g * .c)，加上调试信息；另外，如果使用 Makefile 文件，还可以在 makefile 中定义 CFLAGS 变量，格式如下：

CFLAGS=-g

(1)GDB 的基本用法和选项。

运行 gdb 调试程序时通常使用如下的命令：

```
gdb progname
```

在 gdb 提示符处键入 help，将列出命令的分类，主要的分类有：

①aliases：命令别名。

②breakpoints：断点定义。

③data：数据查看。

④files：指定并查看文件。

⑤internals：维护命令。

⑥running：程序执行。

⑦stack：调用栈查看。

⑧statu：状态查看。

⑨tracepoints：跟踪程序执行。

另外，如果键入 help 后跟命令的分类名，可获得该类命令的详细清单。

(2)GDB 常用命令。

常用的 gdb 命令及解释如下所示：

①break NUM：在指定的行上设置断点。

②bt：显示所有的调用栈帧。该命令可用来显示函数的调用顺序。

③clear：删除设置在特定源文件、特定行上的断点。其用法为 clear FILENAME：NUM。

④continue：继续执行正在调试的程序。该命令用在程序由于处理信号或断点而导致程序停止运行时。

⑤display EXPR：每次程序停止后显示表达式的值。表达式由程序定义的变量组成。

⑥file FILE：装载指定的可执行文件进行调试。

⑦help NAME：显示指定命令的帮助信息。

⑧info break：显示当前断点清单，包括到达断点处的次数等。

⑨info files：显示被调试文件的详细信息。

⑩info func：显示所有的函数名称。

⑪info local：显示当前函数中的局部变量信息。

⑫info prog：显示被调试程序的执行状态。

⑬info var：显示所有的全局和静态变量名称。

⑭kill：终止正被调试的程序。

⑮list：显示源代码段。

⑯make：在不退出 gdb 的情况下运行 make 工具。

⑰next：在不单步执行进入其他函数的情况下，向前执行一行源代码。

⑱print EXPR：显示表达式 EXPR 的值。

⑲Shell shell 命令：不退出 gdb 运行 Shell 命令。

16.5 习题与练习

1. 查看本机 Linux 内核，将其升级至最新版本如果已经是最新版本，则按升级步骤进行内核重建（借助网络工具查找最新内核）。

2. 用 C 语言编写整形数组冒泡排序算法，能够在屏幕上打印排序前后的结果。要求用 vi 编辑器完成源代码编写，用 GCC 和 GDB 工具完成编译和调试。